Troubleshooting B

Troubleshooting is the most time-consuming part of writing any program. This appendix describes how to solve some common problems you might have while using the THINK Project Manager. Among the problems discussed are:

- Crashing
- Bomb alerts, like the one in Figure B-1
- The “out of memory” bug alert
- Unexpected function results

This appendix does not discuss bug alerts with error messages, like the one in Figure B-1. Appendix C, “Error Messages,” describes these messages and suggests how to fix the errors. This appendix does discuss how to avoid bomb alerts, like the one in Figure B-2, and the “out of memory” bug alert.

Figure B-1 Bug alert, explained in Appendix C, “Error Messages”

Figure B-2 Bomb alert, explained in this appendix

Contents

How to Find Your Solution 699

First Try This... 699
Check for virus infection 699
Investigate virus alerts 700
Check for INIT conflicts 700

Launching and Opening Projects 701
Low memory 702
Missing resource error 703

Compiling a Project 703
Corrupted resource file 704

Crashing While Running Your Project. 704
Pointer errors 705
Stack errors 706

Getting Unexpected Function Results 706
Using your own functions 707
Using the ANSI library 707
Using Macintosh Toolbox functions 708
Using math functions 709

Using the Debugger 709

When All Else Fails... 711

How to Find Your Solution

To find the solution to your problem, start with the section “First Try This…” below. It describes how to solve some problems that can arise no matter what you’re doing.

If those solutions don’t work, determine when your problem occurs and refer to the appropriate section in this appendix:

- “Launching and Opening Projects”
- “Compiling a Project”
- “Crashing While Running Your Project”
- “Getting Unexpected Function Results”
- “Using the Debugger”

If you can’t find a solution in the appropriate section, try reading the other sections. Sometimes, you don’t notice a problem until well after its cause has passed.

As a last resort, read the section “When All Else Fails…” later in this appendix. It tells you how to contact Symantec Technical Support.

First Try This…

This section describes how to solve the most common problems that can show up in the THINK Project Manager.

- Check for virus infection
- Investigate virus alerts
- Check for INIT conflicts

Check for virus infection

Viruses cause a large number of unpredictable problems that you can easily confuse with programming errors. If you program frequently, you need antiviral software, like SAM (Symantec Antivirus for Macintosh), Virex, or Disinfectant. Be sure to read the documentation and use the software diligently.

A common virus is `nVIR`, especially in computer labs or any place where many people use one machine. The `nVIR` virus is easy to find without special software. Open your THINK Project Manager application with ResEdit and look for any `nVIR` resources. If you find any, get an antivirus program to eliminate it.

Viruses frequently cause the following problems:

- A “Don’t Panic” dialog, which is a classic symptom of the `nVIR` virus.
- Extra beeping when launching or quitting an application is also an `nVIR` symptom.
- Crashing when you quit an application is another `nVIR` symptom.
- Getting a bomb alert with ID=-39 is an `nVIR` symptom.
- The “out of memory” bug or bomb alert sometimes appears because viruses take up memory and do nothing else. If you can’t find any other possible causes, look for a virus.

Investigate virus alerts

If your antivirus software detects suspicious activities, it will alert you frequently when you use the THINK Project Manager. The THINK Project Manager creates code resources from your source files and adds them to files, usually applications and desk accessories you’re creating. Adding code resources is an activity that antivirus software finds suspicious.

If your antivirus software tells you that the THINK Project Manager is adding or changing a resource to a file, check for a virus. If you have no virus, continue what you were doing and let the THINK Project Manager make the change. If you can teach your antivirus software to let some programs make changes, you’ll want to teach it to let the THINK Project Manager add and change resources.

Check for INIT conflicts

Poorly behaved INITs (Extensions and Control Panels) also cause a large number of unpredictable problems that are easily confused with programming errors.

If your program has a problem that you can’t find in your code, reboot your Macintosh without any INITs. You can do this three ways:

- If you’re running System 7, go to the Finder, choose **Restart** from the **Special** menu, hold down the Shift key as the machine reboots.

- Remove all the INITs from your system folder and reboot.
- Create a new system folder, put the System and Finder from your old system folder in it, and reboot.

If the problem disappears, you know an INIT helped cause it.

To find a troublesome INIT, remove half your INITs and reboot. If there's no problem, you know the INIT is in the half you removed. If there still is a problem, you know the INIT is in the half you let remain. Split the half with the troublesome INIT again and continue this process until you find it.

Old versions of these INITs frequently cause problems with the THINK Project Manager. If you have any problems, make sure you're using the most recent versions of the following:

- MacroMaker
- Automac
- MasterStrokes
- SuperClock
- Easy Access
- Radius FPD and Radius TPD
- QuickFolder
- Staircase
- Stepping Out II

Launching and Opening Projects

This section describes how to solve problems that happen when you launch the THINK Project Manager or open a project, either by choosing **Open Project** in the THINK Project Manager or by double-clicking a project in the Finder.

Look for your problem in this list, and read its suggested solution. If you can't find your problem here, try going through all the solutions.

- Crashing. Read "Low memory" later in this section.
- The "out of memory" bug alert. Read "Low memory" later in this section.
- An error alert with ID=-192. Read "Missing resource error" later in this section.

- An error alert with ID=-39. Read “Check for virus infection” earlier in this appendix.
- A “Don’t Panic” alert. Read “Check for virus infection” earlier in this appendix.
- Extra beeping. Read “Check for virus infection” earlier in this appendix.

Low memory

If you’re using the THINK Debugger, see “Memory Considerations” in Chapter 12. Otherwise, you may have too small a partition for the THINK Project Manager or a corrupted project file. Try these solutions:

- Change the THINK Project Manager’s partition size. When you work with a large project, the THINK Project Manager may need more memory than it usually does. To change the partition size, select the THINK Project Manager in the Finder, choose **Get Info** from the **File** menu, and edit the value in the box labeled Preferred Size (or Application Memory Size in System 6).
- Recompile your project. If you can’t compile or run your project, its object code may be corrupted. Choose **Remove Objects** from the **Project** menu and recompile your project.
- Rebuild your project. Sometimes, a project with bugs accidentally writes to its project file and corrupts it. If you can’t open your project, try opening another one. If it does open, rebuild your project from scratch, creating a new project file and adding the source files.

Note

If you have AppleScript, use the scripts `Save Project As Text` and `Create Project from Text` in the `Scripts` folder.

- Re-install the THINK Project Manager. A project with bugs may damage your THINK Project Manager application. If you think your THINK Project Manager application is damaged, re-install the THINK Project Manager on a dif-

ferent disk. If you can get your work done with the new copy, use it instead of the old one.

Missing resource error

The Macintosh system errors -192 mean that the THINK Project Manager couldn't find a resource. Here are some solutions for that problem:

- Use libraries made with the latest version of THINK C. Don't use a project or library, such as `MacHeaders`, `MacTraps`, and `ANSI`, made with an old version of THINK C. If you are using your own version of `MacHeaders`, precompile it again. Otherwise use the new versions of these libraries supplied with THINK C.

- Use the latest version of the THINK Project Manager to open projects made with the latest version of THINK C. If you have two versions of the THINK Project Manager, you might accidentally launch the old version when you double-click a project made with the new version. Remove the old version, or double-click the new version and open your project there.

- Re-install GateKeeper. If you use the GateKeeper INIT, it may be improperly installed. Re-install it according to its instructions.

- Don't move the THINK Project Manager when it's running. If you move the THINK Project Manager into a different folder while it's running, it won't be able to find its files.

Compiling a Project

This section describes solutions to some problems that happen when you compile a project. This section does not explain the error messages that appear in bug alerts. They're explained in Appendix C, "Error Messages."

Look for your problem in this list, and read its suggested solution. If you can't find your problem here, see Appendix C, "Error Messages."

- A bomb alert with ID=-192. Read "Missing resource error", earlier in this appendix.

- A bomb alert with ID=-39. Read “Check for virus infection” earlier in this appendix.
- Crashing while building an application. Read “Corrupted resource file” below.

Corrupted resource file

If you think your resource file may be corrupted, try these solutions:

- Let ResEdit fix it. Open the resource file with ResEdit, which tries to fix corrupted resource files.
- Use a released version of ResEdit. Be sure to use the ResEdit included with the THINK Project Manager. Sometimes, beta versions of ResEdit contain bugs.
- Rebuild it. If ResEdit can’t fix your file, rebuild it from scratch.

Crashing While Running Your Project

This section describes how to fix problems that cause your program to crash or display a bomb alert.

Look for your problem in this list, and read its suggested solution in this section. If you can’t find your problem here, look through all the solutions.

- A “bus error” (ID=1) bomb alert. You tried to access an address that doesn’t exist. Read “Pointer errors” later in this appendix.
- An “address error” (ID=2) bomb alert. You tried to access memory illegally (that is, you tried to access a word or longword on an odd-byte boundary). Read “Pointer errors” later in this appendix.
- A “stack collision with heap” (ID=28) bomb alert. Your program’s stack became too large and started to overwrite your program’s heap. Read “Stack errors” and “Pointer errors” later in this appendix.
- An “out of memory” (ID=25) bomb alert. Your program ran out of memory. Read “Pointer errors” and “Stack errors” later in this appendix.

- Damaging your hard disk. You may have written to the I/O address space and corrupted your disk. Read “Pointer errors” below.

- Crashing when you call a function the first time. You may not have enough memory to load in the function’s segment. Read “Pointer errors” and “Stack errors” below.

- Crashing when you call a function you called before. You may have used an uninitialized pointer that contains the return address for the function. Writing to the pointer corrupts the function. Read “Pointer errors” below.

Pointer errors

Using pointers improperly can cause a large number of problems. If you’re having problems with pointers, try these solutions:

- Initialize pointers. When you declare a pointer, THINK C doesn’t initialize it or allocate space for what it points to.

- Check array bounds. THINK C doesn’t check whether your array subscripts are within the array bounds.

- Check return values. A function that allocates memory or loads a resource may fail and return null. If you use the pointer, you might corrupt low memory.

- Check your resource file. If a function that loads a resource returns null, maybe it couldn’t find the resource file. Make sure your resource file is in the same folder as your project and has the same name as your project with `.rsrc` appended.

- Increase your program’s partition size. If a function that allocates memory returns null, your program may really be out of memory. Choose **Set Project Type** to see if your partition is large enough for your memory requests.

- Prototype your functions. You might be passing an argument that’s not a pointer to a function that expects a pointer. If you prototype your functions, the THINK Project Manager displays a bug alert when you make this mistake. To make sure you include prototypes, turn on the Strict Prototype Enforcement option and choose Re-

quire prototypes in the Languages Settings page of the **THINK Project Manager options** dialog.

- Check your arguments. You might pass the wrong argument to a Toolbox routine. For example, calling `DisposHandle()` with the handle to a resource crashes your program. You should use `ReleaseResource()`.

- Pass pointers to `var` parameters. If a Pascal function has a `var` parameter, pass it a pointer to the argument.

Stack errors

When your application's stack gets too big, it starts overwriting the application's heap. Every sixtieth of a second, the Macintosh Operating System checks for a stack overflow. If one occurs, it brings up a "stack collision with heap" (ID=28) bomb alert. But you may not get a clear sign that the stack is overflowing. Before the Operating System can catch the overflow, it might cause another problem or even crash your project with a "bus error" (ID=1) or "address error" (ID=2) bomb alert.

If you think your stack might be overflowing, try one of these solutions:

- Use less stack space. Use fewer or smaller local variables. Allocate large variables on the heap.

- Increase the stack size. Use `SetApplLimit()` (in *Inside Macintosh II*) to give your project a larger stack.

Getting Unexpected Function Results

This section describes some reasons why functions don't always return what you expect. To find a solution to your problem, read the parts of the section that deal with the function you're having trouble with:

- "Using your own functions"
- "Using the ANSI library"
- "Using Macintosh Toolbox functions"
- "Using math functions"

If you can't find a solution in the appropriate part, read "Pointer errors" earlier in this appendix.

Using your own functions

If one of your own functions isn't returning what you expected, try this solution:

- Prototype your functions. You might be passing an argument that's not the right type. If you prototype your functions, the THINK Project Manager displays a bug alert when you make this mistake. To make sure you include prototypes, turn on the Strict Prototype Enforcement option and choose Require prototypes in the Languages Settings page of the **THINK C options** dialog.

Using the ANSI library

If you have trouble with any ANSI library functions, try these solutions:

- Don't profile ANSI. Many functions in ANSI won't work if the library is profiled.
- Include the right header file. If you don't include the right header file, THINK C may let you call your function but return invalid results.
- Reinstall the ANSI library. If all else fails (including the solutions below), you may have a corrupted ANSI library. Reinstall it from your original disks.

If you have trouble with the ANSI I/O functions, try these solutions:

- Check the result of `fopen()`. Make sure it's not null.
- Include `stdio.h`. If you don't include `stdio.h`, THINK C may let you call the functions but return invalid results. Even if you don't call any I/O functions, you need to include `stdio.h` if you declare any variables of type `FILE` or `FILE*`.

If you have trouble with the ANSI memory allocation functions, try these solutions:

- Include `stdlib.h`. If you don't include `stdlib.h`, THINK C may let you call the functions but return invalid results.

- Check your return values. Make sure the allocation function doesn't return null.

- Double-check your arguments. Make sure that the arguments that should be of type `size_t` are `size_t`. Remember that multiplying two integers gives you an integer, not a long integer, even if the result doesn't fit in an integer. To make the result a long integer, cast one of the integers to be a long integer.

- Increase your application's partition size. If a memory allocation function returns null, you might really be out of memory. Choose **Set Project Type** to see if your partition is large enough for your memory requests.

- Use the right format specifier in `printf()`. When you print the value of a pointer, use either the pointer specifier `%p`, or one of the `long int` specifiers, such as `%ld` or `%lx`.

Using Macintosh Toolbox functions

Failing to initialize the Macintosh Toolbox managers is a common source of problems. This function takes care of all the initializations most programs need:

```
void InitMac(void)
{
   InitGraf( &thePort );
   InitFonts();
   FlushEvents( everyEvent, 0 );
   InitWindows();
   InitMenus();
   TEInit();
   InitDialogs(0L);
   InitCursor();
   MaxApplZone();
}
```

If you still have trouble with the Toolbox functions, try these solutions:

- Double-check your arguments. Make sure that all the parameters are the correct size, of the correct type, and in the correct order. Remember that a Pascal function with a `var` parameter expects a pointer to the argument.

- Check the return values. Check whether functions that allocate memory or get resources return null. Check whether any function returns an error code. These error codes are described with the functions in *Inside Macintosh*.

- Don't use `AddResFile()`. Instead, use a resource file in the same folder as your project. If you use `AddResFile()` while running under the THINK Project Manager, you corrupt your project's header and you must rebuild your project from scratch.

- Fix your resource file. You may have a corrupted resource file. Open it with ResEdit, which tries to fix corrupted resource files. If ResEdit can't fix it, rebuild it.

- Make sure you add all the resource files you need to your project.

- Check your resource ID numbers. Make sure the ID numbers in the resource file and the ID numbers in your source code match.

- Clean up before sublaunching. Before you launch another application from your application, call `_exiting()` (in the ANSI library).

Using math functions

If you have trouble printing the result of any math function, try this:

- Use the right format specifier in `printf()`. When you print floating-point numbers, use `%g` for the best results. Remember that `%f` rounds to six places after the decimal, so small values print as `0`.

Using the Debugger

This section describes how to solve problems with the THINK Debugger. For more information on how to use the debugger, see Chapter 12, "Debugging Programs." If you're having trouble with low memory, see the "Memory Considerations" in Chapter 12.

- Use System 7 (or MultiFinder under System 6).

- Make sure you have enough memory. For a small project, you need 4MB of RAM to use the debugger. For a large project, you may need more.
- Choose **Use Debugger**. The **Use Debugger** option in the **Project** menu must have a check mark by it.
- Check where the debugger is. The debugger must be in the same folder as the THINK Project Manager.
- Make sure there's debugger information for `main()`. Make sure that there is a gray diamond next to the file that contains `main()` and any file you want to debug.
- Turn off the Use Second Screen option. The debugger may be displayed on a second monitor that doesn't exist or is turned off. Turn off the Use Second Screen option in the Debugging page of the **THINK Project Manager options** dialog.
- Use a smaller partition for your program. If you don't have enough memory for the THINK Project Manager, the debugger, and your project, you may be able to use a smaller partition for your project. Choose **Set Project Type** to see if your partition is too big.
- Use a precompiled header. Using a precompiled header helps make the debugger tables smaller. If you already use the `MacHeaders` file, you can put more header files in it.
- Check your INITs. Some INITs interfere with the debugger's display. Remove them, reboot your Macintosh, and try again. See "Check for INIT conflicts" earlier in this appendix.
- Turn off debugging for some files. If there's not enough memory for the debugger, turn off debugging for some files.

When All Else Fails…

If none of the solutions in this appendix solves your problem, you may need to contact Symantec Technical Support. Before you do, make sure you do the following:

- Isolate your problem. Finding a bug in a large program is difficult. If you think you've found the code causing your problem, write a small program that uses that code.
- Know the System version. To find it, go to the Finder and choose **About the Finder** in the **Apple** menu.
- Know the THINK Project Manager version. To find it, launch the THINK Project Manager and choose **About the THINK Project Manager** in the **Apple** menu.
- Call from a phone near the computer. The support analyst may need to ask you about the program.
- Have the problem in front of you. Be ready to recreate your problem for the support analyst. Have the THINK Project Manager running and your project open.
- Be patient.

To find out how to contact Technical Support, read the "Symantec Customer Service Plan" at the very end of this book.

B Troubleshooting